1-1-2014

Two-party (blind) ring signatures and their applications

Man Ho Au
University of Wollongong, aau@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Two-party (blind) ring signatures and their applications

Two-Party (Blind) Ring Signatures and Their Applications

Man Ho Au and Willy Susilo*

Centre for Computer and Information Security Research (CCISR)
School of Computer Science and Software Engineering
University of Wollongong, Australia
{aau, wsusilo}@uow.edu.au

Abstract. Ring signatures, introduced by Rivest, Shamir and Tauman, attest the fact that one member from a ring of signers has endorsed the message but no one can identify who from the ring is actually responsible for its generation. It was designed canonically for secret leaking. Since then, various applications have been discovered. For instance, it is a building block of optimistic fair exchange, destinated verifier signatures and ad-hoc key exchange. Interestingly, many of these applications require the signer to create a ring signature on behalf of two possible signers (a two-party ring signature) only. An efficient two-party ring signature scheme due to Bender, Katz, and Morselli, is known. Unfortunately, it cannot be used in many of the aforementioned applications since it is secure only in a weaker model. In this paper, we revisit their construction and proposed a scheme that is secure in the strongest sense. In addition, we extend the construction to a two-party blind ring signature. Our proposals are secure in the standard model under well-known number-theoretic assumptions. Finally, we discuss the applications of our construction, which include designated verifier signatures, optimistic fair exchange and fair outsourcing of computational task.

1 Introduction

The notion of ring signatures, introduced by Rivest, Shamir and Tauman [14], is a group-oriented signature which takes into account privacy concerns. A user can autonomously sign on behalf of a group, while group members can be totally unaware of being included in the group. Any verifier can be assured that a message has been endorsed by one of the members in this group, but the actual identity of the signer remains hidden. Unlike group signatures [6], there is no group manager and no revocation. Following the terminology, the group is usually called a ring since the original proposal arrange the group members in a ring during the process of signature generation. The formation of the ring is spontaneous and the original motivation is for Whistle Blowing.

If the ring consists of two members only, the resulting signature is called a two-party ring signature, which is the focus of this paper. The reason is that most of the applications of ring signatures only require a ring size of 2. Nonetheless, even with this

* This work is supported by ARC Future Fellowship FT0991397.

relaxation, construction in the standard model under well-established assumption still remains daunting.

Chow et al. [8] gave a construction with a formal security analysis in the standard model under a new assumption. The general version of Bender et al. [2] uses generic ZAPs for NP as a building block and is inefficient. They also proposed an efficient two-party ring signature scheme under standard assumptions in a weaker security model. Shacham and Waters [17] proposed an efficient ring signature scheme without using random oracles but anonymity is computational and required a trusted setup assumption. Chandran et al. [4] presented a scheme in the untrusted common reference string model while providing "heuristically statistical" anonymity. Schäge and Schwenk [16] provided another ring signature scheme in the standard model using basic assumptions, again, in a weaker security model. Recently, Ghadafi [9] offered both ring signatures and blind ring signatures in the standard model. Again, the anonymity is computational and required a trusted setup assumption.

As mentioned, the focus of our paper is on two-party ring signatures. We start from the specific construction due to Bender et al. due to its simplicity and efficiency. All other existing schemes employ non-interactive proof of some sort in the signature and it is hard to improve their efficiency. In addition, due to the use of non-interactive proof, the anonymity is often computational due to the use of the Groth-Sahai proof system [11] which requires a trusted setup for the common reference string. The major issue with Bender et al.'s two-party ring signature scheme is its insecurity under the chosen key model where an attacker is allowed to generate its own public key. An attack of this kind in this model is presented in [17]. This limits the use of this scheme as a building block in a larger system involving multiple users who could be malicious and are allowed to generate their own keys.

We observe that the problem can be solved if all the keys introduced by the attackers are "certified" to be properly generated. In other words, if we require all users to make a zero-knowledge proof-of-knowledge of the secret keys, the system could be proven secure. For if this is the case, the simulator in the security proof could "extract" the secret key for each public key presented by the adversary and uses this key to answer the query. The only remaining issue is that the prove has to be non-interactive so that it could be viewed as part of this public key. In this regard, we make use of the proof system from Groth-Sahai and present two useful non-interactive protocols as building blocks. The advantage of our approach is that these complex non-interactive proof are only added as part of the public key which only needs to be verified once in practice. Thus, our proposal retains most of the computational advantage of the two-party ring signature scheme from Bender et al. Finally, we equip our scheme with a blind signature generation protocol. We believe our construction can be used as a building block for many larger systems.

1.1 Our Contribution

1. We present efficient non-interactive zero-knowledge proof-of-knowledge protocols for discrete logarithm and commitment.
2. We present an efficient two-party ring signature construction in the standard model and equip it with a blind signing procedure.

3. We discuss various applications of two-party (blind) ring signatures.

Organization. The rest of the paper is organized as follows. In Sec. 2, we review the syntax of a two-party (blind) ring signature scheme its security definitions. We present two non-interactive zero-knowledge proof-of-knowledge protocols in Sec. 3. Based on our protocols, we present an efficient construction of two-party ring signatures and its blind version in Sec. 4. We discuss various applications of our proposal in Sec. 5 before concluding our paper in Sec. 6.

2 Preliminary

If n is a positive integer, we use $[n]$ to denote the set $\{1, \ldots, n\}$. We review the following well-known computational assumptions.

Definition 1 (DL Assumption). *Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p. The discrete logarithm assumption states that given a tuple $(g, Z) \in (\mathbb{G}, \mathbb{G})$, it is computationally infeasible to compute the value $z \in \mathbb{Z}_p$ such that $Z = g^z$.*

Definition 2 (CDH Assumption). *Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p. The computational Diffie-Hellman assumption states that given a tuple $(g, g^a, g^b) \in (\mathbb{G}, \mathbb{G}, \mathbb{G})$, it is computationally infeasible to compute the value g^{ab}.*

Definition 3 (DLIN Assumption). *Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p. The decision linear (DLIN) assumption states that given a tuple $(u, v, w, u^a, v^b, T) \in (\mathbb{G}, \mathbb{G}, \mathbb{G}, \mathbb{G}, \mathbb{G}, \mathbb{G})$, it is computationally infeasible to decide if $T = w^{a+b}$.*

2.1 Bilinear Pairing

Let $\mathbb{G}, \mathbb{G}_T$ be two cyclic groups of the same prime order p. Let g be a generator of $\mathbb{G}$. A mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear pairing if the following are true:

- (*Unique Representation*) Each element of $\mathbb{G}$ and $\mathbb{G}_T$ has a unique binary representation.
- (*Bilinear*) For all $a, b \in \mathbb{Z}_p$, $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$.
- (*Non-degenerate*) $\hat{e}(g, g) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the identity element of the group $\mathbb{G}_T$.

The setting we present here is often referred to as a symmetric pairing. We abuse the notation and use 1 to represent the identity element regardless of the group. For example, we will use 1 to represent $1_{\mathbb{G}_T}$ in the subsequent text.

2.2 Groth-Sahai Non-Interactive Witness-Indistinguishable Proof System

We briefly review the non-interactive witness-indistinguishable proof system developed by Groth and Sahai [11] (referred to as GS proof hereafter). They gave three instantiations and we employ the version that depends on the decision linear assumption since

it is often regarded as the weakest assumption amongst the three and works in the symmetric pairing setting.

Consider a set of variables $\{X_i\}_{i=1}^n \in \mathbb{G}^n$ and public constants $\{A_i\}_{i=1}^n \in \mathbb{G}^n$, $\{b_{i,j} \in \mathbb{Z}_p\}_{i,j\in[n]}$, $t_T \in \mathbb{G}_T$. A pairing product equation is of the form:

$$\prod_{i=1}^{n} \hat{e}(X_i, A_i) \prod_{i=1}^{n} \prod_{j=1}^{n} \hat{e}(X_i, X_j)^{x_{i,j}} = t_T.$$

The GS proof system requires a common reference string which allows a prover to make commitments of a set of variables $\{X_i\}$. It also allows the prover to produce an non-interactive proof that these committed variables satisfy a set of pairing product equations. The proof is witness-indistinguishable in the sense the the proof generated by one set of variables is indistinguishable to another. There are two kinds of common reference string in a GS proof system, namely, soundness string and simulation string. The former allows the string creator to open the "commitments" and thus guarantees the soundness of the system. The latter provides perfectly hiding commitments and guarantees witness-indistinguishably. These two strings are computationally indistinguishable.

As an example, consider a variable X and a constants A and the following pairing product equation:

$$\hat{e}(X, X)\hat{e}(X, A) = 1.$$

A GS proof of a variable X satisfying the above equation could be produced using $X = 1$ or $X = A^{-1}$ and the proofs produced by these two values (often called witnesses) are indistinguishable. Indeed, proof of satisfaction of the above equation would assure the verifier that either $X = 1$ or $X = A^{-1}$.

Throughout this paper, we will use the following notation to represent a GS proof of knowledge of variables satisfying a set of pairing product equations. All symbols that appear on the left hand side inside the brackets are variables while all other symbols on the right hand side after the colon are the public constants of the pairing product equations. For example,

$$NIWI\left\{(X_1, X_2, X_3) : \begin{array}{c} \hat{e}(X_1, X_1)\hat{e}(X_1, A) = 1 \wedge \\ \hat{e}(X_1, X_2)\hat{e}(X_2, X_3) = T \end{array}\right\}$$

means a proof of the variables X_1, X_2, X_3 that satisfies the two equations, with A and T being public constants.

2.3 Syntax of Two-Party Ring Signatures

We adapt the definitions and security models of ring signatures from various literatures. A ring signature scheme in the common reference string model consists of four algorithms, namely, Setup, Gen, Sign, Verify, whose functions are enumerated below.

$\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$: On input a security parameter λ, this algorithm outputs the public parameter param for the system. We assume param is an implicit input to all algorithms listed below.

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}()$: This algorithm outputs a key pair $(\mathsf{pk}, \mathsf{sk})$ for a signer. If $(\mathsf{pk}, \mathsf{sk})$ is an output of the algorithm $\mathsf{Gen}()$, we say pk is the corresponding public key of sk (and vice versa).

$(\sigma, \mathcal{R}) \leftarrow \mathsf{Sign}(\mathsf{sk}_S, \mathcal{R}, m)$: On input a message m, a secret key of a signer sk_S (whose public key is pk_S) and a set public keys $\mathcal{R}$ with $\mathsf{pk}_S \in \mathcal{R}$, this algorithm outputs a signature σ, which is a ring signature of m with respect to the ring $\mathcal{R}$.

$\texttt{valid/invalid} \leftarrow \mathsf{Verify}(\sigma, \mathcal{R}, m)$: On input a public key a message m, a signature σ with a set of public keys $\mathcal{R}$, this algorithm verifies the signature and outputs $\texttt{valid/invalid}$.

A ring signature scheme must possess *Correctness*, *Unforgeability* and *Anonmity*, to be reviewed below.

Correctness. For any security parameter λ and $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{Gen}()$ and $\mathcal{R} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ such that $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}()$ for $i \in [n]$ with $\mathsf{pk}_S \in \mathcal{R}$. For any message m, if $(\sigma, \mathcal{R}) \leftarrow \mathsf{Sign}(\mathsf{sk}_S, \mathcal{R}, m)$, then $\texttt{valid} \leftarrow \mathsf{Verify}(\sigma, \mathcal{R}, m)$.

In this paper, our focus is on two-party ring signature. That is, $|\mathcal{R}| = 2$ for all signatures.

Unforgeability. The following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally captures the requirement of *Unforgeability*.

Setup $\mathcal{C}$ invokes $\mathsf{Setup}(1^\lambda)$ and subsequently $\mathsf{Gen}()$ to obtain $(\mathsf{param}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]})$. Denote the set $\{\mathsf{pk}_i\}_{i\in[n]}$ by $\mathcal{R}$. $(\mathsf{param}, \mathsf{pk}_S, \mathcal{R})$ is given to $\mathcal{A}$.

Query $\mathcal{A}$ is allowed to make the following queries:

- Corruption Query. $\mathcal{A}$ submits a public key $\mathsf{pk}_i \in \mathcal{R}$ and receives sk_i.
- Signature Query. $\mathcal{A}$ submits a message m, an arbitrary ring $\mathcal{R}' = \{\mathsf{pk}'_0, \mathsf{pk}'_1\}$ and a bit b, and receives $(\sigma, \mathcal{R}') \leftarrow \mathsf{Sign}(\mathsf{sk}'_b, \mathcal{R}', m)$ where sk'_b is the corresponding private key of pk'_b.

Output $\mathcal{A}$ submits $(\sigma^*, \mathcal{R}^*, m^*)$ and wins if and only if

1. $\texttt{valid} \leftarrow \mathsf{Verify}(\sigma^*, \mathcal{R}^*, m^*)$ and $\mathcal{R}^* \subset \mathcal{R}$.
2. $\mathcal{A}$ has not submitted a Signature Query with input m^*, $\mathcal{R}^*$.
3. $\mathcal{A}$ has not submitted a Corruption Query on input pk such that $\mathsf{pk} \in \mathcal{R}^*$.

Definition 4 (Unforgeability). *A two-party ring signature scheme is unforgeable if no PPT adversary wins the above game with non-negligible probability.*

Our definition of unforgeability is slightly stronger than the strongest notion, existential unforgeability with respect to insider corruption [2], in which we allow the adversary to issue signature query on behalf of arbitrary ring without supplying the corresponding secret key.

Anonymity. It means that given a message m and a signature $(\sigma, \mathcal{R})$, it is infeasible to determine who created the signature, even if all the secret keys are known. The formal definition is adapted from [2] for general ring signatures against full key exposure. Note that we allow the common reference string to be maliciously generated in this model.

The following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally captures the requirement of *Anonymity*.

Setup $\mathcal{A}$ gives param to $\mathcal{C}$. $\mathcal{C}$ invokes $\mathsf{Gen}()$ to obtain $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]})$. Denote the set $\{\mathsf{pk}_i\}_{i\in[n]}$ by $\mathcal{R}$. (pk_i, sk_i) for $i \in [n]$) is given to $\mathcal{A}$.

Challenge $\mathcal{A}$ gives two indexes i_0, i_1 and a message m to $\mathcal{C}$. $\mathcal{C}$ filps a fair coin b and computes $(\sigma, \{\mathsf{pk}_{i_0}, \mathsf{pk}_{i_1}\}) \leftarrow \mathsf{Sign}(\mathsf{sk}_b, m\{\mathsf{pk}_{i_0}, \mathsf{pk}_{i_1}\})$. σ is returned to $\mathcal{A}$.

Output $\mathcal{A}$ submits a guess bit b' and wins if and only if $b' = b$.

Definition 5 (Anonymity). *A two-party ring signature is unconditionally anonymous if no computationally unbounded adversray $\mathcal{A}$ wins the above game with probability that is non-negligibly higher than $1/2$.*

2.4 Syntax of Two-Party Blind Ring Signatures

A signature is blind if there exists a protocol between the signer and a user in which the user obtains a signature from the signer on message m in such a way that the signer learns nothing about the m nor the signature issued. More formally, a blind signature scheme is a scheme with the following additional protocol BSign.

Setup, Gen, Verify are the same as above.

BSign : This is a protocol between the signer and a user. The common input is param and a ring of two public keys $\mathcal{R} = (\mathsf{pk}_0, \mathsf{pk}_1)$. The signer has additionally a private input a private key (sk_b) such that the corresponding public key (pk_b) is in the ring $\mathcal{R}$. The user has a private input m. Upon successful completion of the protocol, the user obtains a ring signature σ on message m with respect to ring $\mathcal{R}$.

Unforgeability. The following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally captures the requirement of *Unforgeability* for any two-party blind ring signatures.

Setup $\mathcal{C}$ invokes $\mathsf{Setup}(1^\lambda)$ and subsequently $\mathsf{Gen}()$ to obtain $(\mathsf{param}, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]})$. Denote the set $\{\mathsf{pk}_i\}_{i\in[n]}$ by $\mathcal{R}$. $(\mathsf{param}, \mathsf{pk}_S, \mathcal{R})$ is given to $\mathcal{A}$.

Query $\mathcal{A}$ is allowed to make the following queries:

- Corruption Query. $\mathcal{A}$ submits a public key $\mathsf{pk}_i \in \mathcal{R}$ and receives sk_i.
- Blind Sign Query. $\mathcal{A}$ submits an arbitrary ring $\mathcal{R}' = \{\mathsf{pk}'_0, \mathsf{pk}'_1\}$ and a bit b, and interacts with $\mathcal{C}$ who plays the role of a signer (with public key pk'_b and secret key sk'_b).

Output $\mathcal{A}$ submits a ring $\mathcal{R}^*$ and $(k+1)$ distinct messages (m^*_i) and their corresponding signatures (σ^*_i) for $i = 1$ to $k+1$ and wins if and only if

1. $\mathcal{R}^* \subset \mathcal{R}$ and $\mathcal{A}$ has not submitted a Corruption Query on input pk such that $\mathsf{pk} \in \mathcal{R}^*$.
2. $\mathtt{valid} \leftarrow \mathsf{Verify}(\sigma^*_i, \mathcal{R}^*, m^*_i)$ for $i = 1$ to $k+1$.
3. $\mathcal{A}$ has submitted at most k blind sign queries with input $\mathcal{R}^*$.

Definition 6 (Unforgeability). *A two-party blind ring signature scheme is unforgeable if no PPT adversary wins the above game with non-negligible probability.*

Anonymity Anonymity for the blind version is shown as follows.

The following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally captures the requirement of *Anonymity*.

Setup $\mathcal{A}$ gives param to $\mathcal{C}$. $\mathcal{C}$ invokes $\mathsf{Gen}()$ to obtain $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]})$. Denote the set $\{\mathsf{pk}_i\}_{i\in[n]}$ by $\mathcal{R}$. $\mathcal{C}$ gives (pk_i, sk_i) for $i \in [n]$ to $\mathcal{A}$.

Challenge $\mathcal{A}$ gives two indexes i_0, i_1 to $\mathcal{C}$. $\mathcal{C}$ filps a fair coin b and interacts with $\mathcal{A}$ as a signer in protocol BSign with private input sk_{i_b} .

Output $\mathcal{A}$ submits a guess bit b' and wins if and only if $b' = b$.

Definition 7 (Anonymity). *A two-party blind ring signature is unconditionally anonymous if no computationally unbounded adversray $\mathcal{A}$ wins the above game with probability that is non-negligibly higher than $1/2$.*

Blindness Blindness refers to the fact that the signer learns nothing about the message being signed nor the signature created during a blind sign protocol. The following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally captures the requirement of *Blindness*.

Setup $\mathcal{C}$ invokes $\mathsf{Setup}()$ and subsequently $\mathsf{Gen}()$ to obtain param, $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in[n]})$. Denote the set $\{\mathsf{pk}_i\}_{i\in[n]}$ by $\mathcal{R}$. $\mathcal{C}$ gives param and (pk_i, sk_i) for $i \in [n]$ to $\mathcal{A}$.

Challenge $\mathcal{A}$ gives a ring $\mathcal{R}^* \subset \mathcal{R}$ and two messages m_0, m_1 to $\mathcal{C}$. $\mathcal{C}$ filps a fair coin b and obtains two signatures from $\mathcal{A}$ in protocol BSign in the following order: the first interaction is to obtain signature on message m_b and the second interaction is for signature on message m_{1-b}. $\mathcal{C}$ gives σ_0, σ_1 to $\mathcal{A}$ which are the resulting signatures from these interactions. Note that the index i is arbitrary, meaning that σ_i could be the result from the first run or the second interaction. Further, $\sigma_i = \perp$ if the interaction does not terminate successfully.

Output $\mathcal{A}$ submits a guess bit b' and wins if and only if $b' = b$.

Definition 8 (Blindness). *A two-party blind ring signature is blind if no PPT adversray $\mathcal{A}$ wins the above game with probability that is non-negligibly higher than $1/2$.*

3 Non-Interative Zero-Knowledge Proof-of-Knowledge

Our construction relies on an non-interactive zero-knowledge proof-of-knowledge of discrete logarithm satisfying the requirement of what is commonly referred to as signature proof-of-knowledge. More formally, it has to be simulatable and extractable under the same common reference string. Being simulatable means that given a trapdoor of the common reference string, there exists an efficient algorithm, called simulator, which is capable of simulating a proof-of-knowledge of discrete logarithm of an element without actually knowing the discrete logarithm. Being extractable means that given the trapdoor of the common reference string, there exists another efficient algorithm, called extractor, which is capable of outputting the discrete logarithm of an element given a proof-of-knowledge of discrete logarithm of that element.

Many secure constructions requires a proof system that allows simulations and extraction under the same string and this is not readily achievable in the GS proof system.

Nonetheless, Bernhard et al. [3], following the technique employed in [10], showed how a GS proof for pairing product equations can be turned into signature of knowledges offering simulatability and extractability simultaneously. Our protocol follows their concept and the difference is discussed after the presentation of the protocol.

3.1 Proof $\mathfrak{P}_{\mathsf{DL}}$

We use the notation $\mathfrak{P}_{\mathsf{DL}}\{(x) : Y = g^x\}$ to represent an non-interactive zero-knowledge proof-of-knowledge of the discrete logarithm of Y to base g. The protocol makes use of the GS witness-indistinguishable proof system on a set of pairing-product equations. Our protocol is inspired by various constructions in the literature and we introduce several optimizations for efficiency considerations.

Setup Let n be the security parameter. The common reference string of $\mathfrak{P}_{\mathsf{DL}}$ consists of $\mathsf{crs}_{\mathsf{GS}}$, the soundness string of the GS proof system, and the following elements $v', v_1, \ldots, v_n, h_1, h_2 \in \mathbb{G}$ and a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$.

Proof Generation Intuition We first present the intuition of the non-interactive proof of knowledge of x such that $Y = g^x$. The prover first express x as $\sum_{i=0}^{n-1} 2^i x[i]$ where $x[n-1] \ldots x[0]$ is the binary representation of x. The prover generates a witness-indistinguishable proof of the following fact:

$$NIWI \left\{ (X_0, \ldots, X_{n-1}, S_1) : \begin{array}{r} \left(\begin{array}{c} \bigwedge_{i=0}^{n-1} (X_i = g^{2^i} \vee X_i = 1) \wedge \\ Y = \prod_{i=0}^{n-1} X_i \qquad \wedge \end{array} \right) \\ \vee \\ \left(\hat{e}(S_1, h) = \hat{e}(h_1, h_2) \hat{e}(\mathcal{V}(statement), S_2) \right) \end{array} \right\}$$

where $\mathcal{V}(statement)$ is the waters hash of the statement being proved. Specifically, let C be the commitments of the witnesses of $\{X_i\}$'s in the GS proof system. Further, let $s = H(C||Y||S_2)$ and $\mathcal{V}(statement)$ is defined as $v' \prod v_i^{s[i]}$ where $s[i]$ is the i-th bit of s. The idea of the proof is that $X_i = g^{2^i}$ if $x[i]$ is 1 and $X_i = 1$ (the identity element of $\mathbb{G}$) if $x[i] = 0$. Due to the soundness of the GS proof system, knowing the set of X_i's implies knowing a set of values $x[i] \in \{0,1\}$. Since the later part assures the verifier that $Y = \prod_{i=0}^{n-1} X_i$, knowing a set of values $x[i] \in \{0,1\}$ is equivalent to knowing the value $x = \sum_{i=0}^{n-1} x[i]$ such that $Y = g^x$. One could view the final equation as knowing a Waters' signature on the statement. In fact, (S_1, S_2) is a Waters' signature on the value $s = H(C||Y||S_2)$. This is purely for the simulatablility of the proof as the simulator who generates the common reference string knows the "secret key" (α) such that $h_1 = h^\alpha$ would be capable of generating (S_1, S_2) and use it for the proof simulation. Due to witness-indistinguishability of the GS proof system, the simulated proof is indistinguishable to the proof generating using witness $\{X_i\}$. The remaining challenge is to transform the above idea into a set of pairing-product equations where the GS proof system can be used.

Proof Generation The prover randomly picks $r \in_R \mathbb{Z}_p$, computes $S_2 = h_1^r \in \mathbb{G}$ and computes the following GS proof.

$$NIWI \left\{ (X_0, \ldots, X_{n-1}, S_1, S) : \begin{array}{rc} \hat{e}(S/h, S/h_1) = 1 & \wedge \\ \bigwedge_{i=0}^{n-1} (\hat{e}(X_i, X_i/g^{2^i}) = 1) & \wedge \\ \hat{e}(Y, S/h) = \hat{e}(\prod_{i=0}^{n-1} X_i, S/h) & \wedge \\ \hat{e}(S_1, S) = \hat{e}(h_2, h_1)\hat{e}(\mathcal{V}(statement), S_2) & \end{array} \right\}$$

using the set of witnesses $\{X_i = g^{x[i]2^i}\}, S = h_1, S_1 = \mathcal{V}(statement)h_2^r$. Again, let C be the commitments of $\{X_i\}$'s and S in the GS proof system. Let $s = H(C||Y||S_2)$ and $\mathcal{V}(statement)$ is defined as $v' \prod v_i^{s[i]}$ where $s[i]$ is the i-th bit of s.

Proof Verification The verifier validates the NIWI proof and that $S_2 \neq 1$.

Discussions Various optimization techniques have been employed for the realization of $\mathfrak{P}_{\mathsf{DL}}$. Firstly, one could note that the variable S acts as a selector. The first equation guarantees that $S = h$ or $S = h_1$. When $S = h_1$, the prover can simulate the last equation without knowing the Waters signature on the statement as shown above. Specifically, the prover can set $S_2 = h_1^r$ and use the witness $S_1 = \mathcal{V}(statement)h_2^r$. In this setting, the prover is forced to set X_i to be $g^{x[i]2^i}$ so that the equation $\hat{e}(Y, h_1/h) = \hat{e}(\prod_{i=0}^{n} X_i, h_1/h)$ holds. Finally, one could note that the equation $\hat{e}(X_i, X_i/g^{2^i}) = 1$ guarantees that $X_i = 1$ or g^{2^i}.

To produce a simulated proof, the simulator would have to generate the common reference string so that it knows α such that $h_1 = h^\alpha$. With this trapdoor (α), the simulator could set $X_i = 1$, $S = h$ so that all but the last equation holds. For the last equation, the simulator creates a Waters signature on the statement. Specifically, it randomly picks $r \in_R \mathbb{Z}_p$, computes $S_2 = h^r$ and $S_1 = h_2^\alpha \mathcal{V}(statement)^r$.

While the general idea is similar to that of [3] in the construction of signature proof of knowledge, our realization contains several optimizations. Firstly, only part of the Waters' signature (S_1) is a variable. The reason is that S_2 can be computed without knowing the signing key and can be "simulated" by the real prover. Secondly, we make use of the symmetric pairing to save the number of variables for the "OR" proofs. For instance, proving the knowledge of variable X satisfying the relation $e(X/A, X/B) = 1$ is equivalent to proving $X = A$ or $X = B$ with one pairing-product equation in one variable.

3.2 Proof $\mathfrak{P}_{\mathsf{WH}}$

Let $g, u_1, \ldots, u_n \in \mathbb{G}$ be generators of group $\mathbb{G}$. Consider a message $m \in \mathbb{Z}_p$, we consider the commitment of m as $\prod_{i=1}^{n} u_i^{m[i]} g^r$ for a random value $r \in_R \mathbb{Z}_p$ where $m[i]$ is the i-th bit of m. We use the notation $\mathfrak{P}_{\mathsf{WH}}\{(m, r) : M = \prod_{i=1}^{n} u_i^{m[i]} g^r\}$ to represent the non-interactive zero-knowledge proof-of-knowledge of how the commitment M can be openned.

Common Reference String is the same as the proof $\mathfrak{P}_{\mathsf{DL}}$.

Proof Generation The prover randomly picks $t \in_R \mathbb{Z}_p$, computes $S_2 = h_1^t \in \mathbb{G}$ and computes the following GS proof.

$$NIWI \left\{ \begin{pmatrix} R_0, \ldots, R_{n-1}, \\ U_1, \ldots, U_n, \\ S_1, S \end{pmatrix} : \begin{array}{rl} \hat{e}(S/h, S/h_1) = 1 & \wedge \\ \bigwedge_{i=1}^{n} (\hat{e}(U_i, U_i/u_i) = 1) & \wedge \\ \bigwedge_{i=0}^{n-1} (\hat{e}(R_i, R_i/g^{2^i}) = 1) & \wedge \\ \hat{e}(M, S/h) = \hat{e}(\prod_{i=1}^{n} U_i \prod_{i=0}^{n-1} R_i, S/h) & \wedge \\ \hat{e}(S_1, S) = \hat{e}(h_2, h_1)\hat{e}(\mathcal{V}(statement), S_2) & \end{array} \right\}$$

using the set of witnesses $\{R_i = g^{r[i]2^i}\}, \{U_i = u_i^{m[i]}\}, S = h_1, S_1 = \mathcal{V}(statement)h_2^t$. Here $r[n-1] \ldots r[0]$ is the binary representation of r. Again, let C be the commitments of $\{R_i\}$'s, $\{U_i\}$'s and S in the GS proof system. Let $s = H(C||M||S_2)$ and $\mathcal{V}(statement)$ is defined as $v' \prod v_i^{s[i]}$ where $s[i]$ is the i-th bit of s.

Proof Verification The verifier validates the NIWI proof and that $S_2 \neq 1$.

Regarding the security of these two non-interactive proofs, we have the following theorem whose proof can be found in Appendix A.

Theorem 1. *$\mathfrak{P}_{\mathsf{DL}}$ and $\mathfrak{P}_{\mathsf{WH}}$ are simulatable under the DLIN assumption and extractable under the CDH assumption in the standard model.*

4 Constructions

We first describe our construction of a two-party ring signature, which is essentially the scheme from [2] with a proof-of-correctness added to the public key.

4.1 A Two-Party Ring Signature Scheme

$\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$: On input a security parameter λ, this algorithm chooses two cyclic groups $\mathbb{G}$, $\mathbb{G}_T$ of prime order p such that $|p| = \lambda$ and that there exists a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. It also generates the common reference string of the non-interactive proof system discussed in section 3. Finally, it chooses several generator $g, u', u_1, \ldots, u_n \in \mathbb{G}$, where n is the bit-length of the message.

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}()$: This algorithm randomly picks $x \in_R \mathbb{Z}_p$ and computes $Y = g^x$. It also computes the non-interactive proof $\pi_Y = \mathfrak{P}_{\mathsf{DL}}\{(x) : Y = g^x\}$. The public key is (Y, π_Y) and the secret key is x.

$(\sigma, \mathcal{R}) \leftarrow \mathsf{Sign}(\mathsf{sk}_S, \mathcal{R}, m)$: On input a message m, a secret key of a signer x and two public keys (Y, π_Y) and $(Y', \pi_{Y'})$ where $Y = g^x$, this algorithm first validates $\pi_{Y'}$. Next, it computes $M = u' \prod_{i=1}^{n} u_i^{m[i]}$. It then chooses $r \in_R \mathbb{Z}_p$, computes $S_2 = g^r$ and $S_1 = Y'^x M^r$. Output σ as (S_1, S_2) and the ring as $\{(Y, \pi_Y), (Y', \pi_{Y'})\}$.

$\texttt{valid/invalid} \leftarrow \mathsf{Verify}(\sigma, \mathcal{R}, m)$: On input a a message m, a signature σ and a rinf $\{(Y, \pi_Y), (Y', \pi_{Y'})\}$, this algorithm first validates π_Y and $\pi_{Y'}$. Next, it outputs `valid` if and only if

$$\hat{e}(S_1, g) = \hat{e}(Y, Y')\hat{e}(u' \prod_{i=1}^{n} u_i^{m[i]}, S_2).$$

Note that the signature size is only 2 elements. While the validations of π_Y and $\pi_{Y'}$ is quite expensive, it is required to be conducted once per each public key. Thus, in a long run, our scheme is nearly as efficient as the original scheme from [2] if the public key of the users are relatively stable.

4.2 A Blind Signature Generation Protocol for our Two-Party Ring Signature Scheme

The blind signature generation protocol is a two-round protocol described below. The public keys of the ring, (Y, π_Y) and $(Y', \pi_{Y'})$ are known to both the signer and the user. Without loss of generality, the signer has an additional input x such that $Y = g^x$. The user has an additional input m.

The user validates (Y, π_Y) and $(Y', \pi_{Y'})$. Then it computes a commitment of m as $M' = \prod_{i=1}^{n} u_i^{m[i]} g^r$ for some random $r \in_R \mathbb{Z}_p$ and the non-interactive proof $\pi_{M'} = \mathfrak{P}_{\mathsf{WH}}\{(m, r) : M' = \prod_{i=1}^{n} u_i^{m[i]} g^r\}$. It sends $M', \pi_{M'}$ to the signer.

Upon receive $M', \pi_{M'}$, the signer validates $\pi_{M'}$ and $\pi_{Y'}$. Next, it computes $S_2 = g^t$ for some randomly generated $t \in_R \mathbb{Z}_p$, $S_1 = {Y'}^x (u'M')^t$ and returns (S_1', S_2') to the user.

The user picks $a \in_R \mathbb{Z}_p$, computes $S_1 = (S_1'/(S_2')^r)(u'M')^a$ and $S_2 = S_2' g^a$. It outputs the signature on m as (S_1, S_2).

It is straightforward to see that the signature created using the blind signature generation protocol has the same distribution as those outputted from the sign algorithm.

Regarding the security of our two-party ring signatures, we have the following theorem whose proof can be found in Appendix B.

Theorem 2. *Our construction of two-party ring signatures is unforgeable under the DLIN and CDH assumption. It is unconditionally anonymous. The blind signature version possesses blindness under the DLIN and CDH assumption.*

5 Applications

Designated Verifier Signatures A direct application of our two-party ring signature scheme is on designated verifier signatures, Jakobsson, Sako and Impagliazzo [13], and independently by Chaum [5] in 1996. A DVS scheme allows a signer Alice to convince a designated verifier Bob that Alice has endorsed the message while Bob cannot transfer this conviction to anyone else. As discussed in [15], if Alice create a ring signature on behalf of the ring with Alice and Bob and sends the ring signature to Bob, Bob will be convinced that the message has been endorsed by Alice. On the other hand, the signature will not be able to convince any outsider since Bob could have been the creater of the signature. Hence, a two-party ring signature is sufficient for the construction of designated verifier signature. Our construction secure in the strong model is necessary for it allows the resulting designated verifier signatures to be secure against the vogue key attack [13, 18]. On the other hand, some existing schemes make use of some ad-hoc techniques [20] to defend against this attack.

Optimistic Fair Exchange Optimistic fair exchange, introduced by Asokan, Schunter and Waidner [1] allows two parties, Alice and Bob, to exchange digital signatures with the help of a passive trusted third party. Haung et al. [12] presents an elegant realization based on a secure two-party ring signatures. Their construction is generic in which any secure two-party ring signature scheme can be used. Having said that, since optimistic fair exchange is supposed to work in the multi-user setting, the security requirement of the underlying ring signature is stronger than the model guaranteed by the two-party ring signatures in [2]. On the other hand, our scheme satisfies their security requirements and can be used. We also make the following observation. If the blind version of our ring signature scheme is employed, the resulting optimistic fair exchange protocol enjoys an additional property in which the trusted third party cannot learn anything about the messages and signatures being exchanged even if it is called upon for protocol completion. This will improve the applicability of this protocol since they exchanging parties might be reluctant to reveal the information of the exchange.

Fair Outsourcing Following the fair exchange paradigm, Chen et al. [7] consider the problem of the exchange of payment and outsourcing computation. In their proposal, the job owner outsourced some computationally expensive task to a set of workers and upon completion of its assigned computation, the worker shall receive the payment from the job owner. A fair exchange protocol is used to ensure fairness. Specifically, the computation result is used in exchange of the job owner's payment. As a two-party ring signature can be used as a building block for a fair exchange protocol, our construction is also useful in the fair outsourcing system.

6 Conclusion

In this paper, we present two useful non-interactive zero-knowledge proof-of-knowledge protocols. With these protocols, we proposed an efficient two-party ring signatures and its extension to support blind signature generation. Finally, we discussed several applications of our constructions.

References

1. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In R. Graveman, P. A. Janson, C. Neumann, and L. Gong, editors, *ACM Conference on Computer and Communications Security*, pages 7–17. ACM, 1997.
2. A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In S. Halevi and T. Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2006.
3. D. Bernhard, G. Fuchsbauer, and E. Ghadafi. Efficient signatures of knowledge and daa in the standard model. In M. J. J. Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS*, volume 7954 of *Lecture Notes in Computer Science*, pages 518–533. Springer, 2013.
4. N. Chandran, J. Groth, and A. Sahai. Ring signatures of sub-linear size without random oracles. In L. Arge, C. Cachin, T. Jurdzinski, and A. Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 423–434. Springer, 2007.
5. D. Chaum. Private Signature and Proof Systems, 1996. US Patent 5,493,614.

6. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.
7. X. Chen, J. Li, and W. Susilo. Efficient fair conditional payments for outsourcing computations. *IEEE Transactions on Information Forensics and Security*, 7(6):1687–1694, 2012.
8. S. S. M. Chow, J. K. Liu, V. K. Wei, and T. H. Yuen. Ring Signatures Without Random Oracles. In *ASIACCS 06*, pages 297–302. ACM Press, 2006.
9. E. Ghadafi. Sub-linear blind ring signatures without random oracles. Cryptology ePrint Archive, Report 2013/612, 2013. http://eprint.iacr.org/.
10. J. Groth. Simulation-sound nizk proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.
11. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.
12. Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Efficient optimistic fair exchange secure in the multi-user setting and chosen-key model without random oracles. In T. Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2008.
13. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and Their Applications. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 1996.
14. R. L. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer, 2001.
15. S. Saeednia, S. Kremer, and O. Markowitch. An Efficient Strong Designated Verifier Signature Scheme. In J. I. Lim and D. H. Lee, editors, *ICISC*, volume 2971 of *Lecture Notes in Computer Science*, pages 40–54. Springer, 2003.
16. S. Schäge and J. Schwenk. A cdh-based ring signature scheme with short signatures and public keys. In R. Sion, editor, *Financial Cryptography*, volume 6052 of *Lecture Notes in Computer Science*, pages 129–142. Springer, 2010.
17. H. Shacham and B. Waters. Efficient ring signatures without random oracles. In T. Okamoto and X. Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2007.
18. K.-A. Shim. Rogue-key Attacks on the Multi-designated Verifiers Signature Scheme. *Inf. Process. Lett.*, 107(2):83–86, 2008.
19. V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.
20. Y. Zhang, M. H. Au, G. Yang, and W. Susilo. (strong) multi-designated verifiers signatures secure against rogue key attack. In L. Xu, E. Bertino, and Y. Mu, editors, *NSS*, volume 7645 of *Lecture Notes in Computer Science*, pages 334–347. Springer, 2012.

A Proof of Theorem 1

We sketch the proof idea for $\mathfrak{P}_{\mathsf{DL}}$. The proof for $\mathfrak{P}_{\mathsf{WH}}$ is similar and is thus omitted.

Simulatability In the intuition of $\mathfrak{P}_{\mathsf{DL}}$, we already discussed how a simulator, with the knowledge of α such that $h_1 = h^{\alpha}$ can produce a simulated proof $\mathfrak{P}_{\mathsf{DL}}$. It remains to argue this simulated proof is indistinguishable from the real proof. The argument makes use of the game-hoping technique [19] which involves a sequence of games defined below.

1. Game_0: This is the real game.
2. Game_1: Same is Game_0 except the common reference string of the GS proof system $\mathsf{crs}_{\mathsf{GS}}$ is chosen in the simulation setting instead of the soundness setting.
3. Game_2: Same as Game_1 except the non-interactive proofs $\mathfrak{P}_{\mathsf{DL}}$ given to the adversary is generated by the simulator.
4. Game_3: Same as Game_2 except the common reference string of the GS proof system $\mathsf{crs}_{\mathsf{GS}}$ is chosen to be in the soundness setting instead of the simulation setting.

The setting in Game_3 is where the adversary is given simulated proofs instead of real proofs. It remains to show the advantage of an adversary trying to distinguish whether it is playing Game_0 and Game_3 is negligible. The argument goes as follows. The difference between Game_0 and Game_1 is negligible under the DLIN assumption due to the computational indistinguishably of the common reference string of the GS proof system. For Game_1 and Game_2, observe that the distribution of S_2 is the same (uniformly at random from $\mathbb{G}$) and that the distribution of the GS proof is also the same (since the commitments are perfectly hiding in the simulation string), the difference between Game_1 and Game_2 is negligible. Finally, the difference between Game_2 and Game_3 is negligible under the DLIN assumption. Thus, $\mathfrak{P}_{\mathsf{DL}}$ is simulatable under the DLIN assumption.

Extractability Due to the soundness of the GS proof system, the extractor can always extracts from $\mathfrak{P}_{\mathsf{DL}}$ a set of witnesses $(X_0, \ldots, X_{n-1}, S_1, S)$ satisfying the set of pairing product equations. From the equation

$$\hat{e}(S/h, S/h_1) = 1,$$

S can only be h or h_1. If $S = h$, the last equation

$$\hat{e}(S_1, S) = \hat{e}(h_2, h_1)\hat{e}(\mathcal{V}(statement), S_2)$$

means that the witness S_1 together with the value S_2 is a Waters signature on the message "statement". Since the adversary must be producing a new statement. It is easy to setup the simulator which breaks the existential unforgeability of Waters signature, which is equivalent to solve the CDH problem. (The simulator is given a signing oracle of the Waters signature and use it to produce all the simulated proof. Finally, we then adversary produces a new proof, the simulator extracts a new Waters signature.)

When $S = h_1$, we have

$$\hat{e}(Y, S/h) = \hat{e}(\prod_{i=0}^{n-1} X_i, S/h),$$

which implies $Y = \prod_{i=0}^{n-1} X_i$. Recall that for all i,

$$\hat{e}(X_i, X_i/g^{2^i}) = 1,$$

it means $X_i = 1$ or $X_i = g^{2^i}$. From this, the simulator can calculate $x = \sum_{i=0}^{n-1} x[i]2^i$ where $x[i] = 0$ if $X_i = 1$ and $x[i] = 2^i$ otherwise.

Thus, $\mathfrak{P}_{\mathsf{DL}}$ is extractable under the CDH assumption.

B Proof of Theorem 2

We sketch the proof idea for our construction of two-party (blind) ring signatures.

Unforgeability The signature generation and verification is the same as the construction of [2], which is unforgeable against chosen sub-ring attack. The only difference is the addition of the non-interactive proof of knowledge of the secret key attached in the public key. For all public keys presented by the adversary, the simulator can extract the corresponding signing keys and use it to answer all queries related to these keys chosen by the adversary. This in turns allow our scheme to be proven secure in the stronger model where the adversary can introduce keys into the system. For the blind signature generation protocol, the simulator can always extract from $\mathfrak{P}_{\mathsf{WH}}$ the message to be signed in the protocol and the rest is the same as the origin version. Due to the need to produce a simulated proof for the challenge public key and the need of extractions, our construction is secure under the CDH and the DLIN assumption in the standard model.

Anonymity The original two-party ring signature of [2] is unconditionally anonymous and it is straightforward to see our construction retains this desirable property. Consider a given signature $(S_1, S_2), \{PK, PK'\}$ on message m. For any public key $PK = (Y, \pi_Y)$, there exists a random value r such that $S_1 = g^{yy'}(u' \prod_{i=1}^{m} u_i^{m[i]})^r$ and $S_2 = g^r$. In other words, it can be generated by user with public key PK or PK' and thus the signature is unconditionally anonymous.

Blindness Consider two transcripts of the blind signature generation protocol (M', $\pi_{M'}$, S'_1, S'_2) and ($\bar{M}'$, $\pi_{\bar{M}'}$, $\bar{S}'_1$,$\bar{S}'_2$) and a given message-signature pair (m, S_1, S_2).

There exists a set of randomness (r, a) such that $M' = \prod_{i=1}^{n} u_i^{m[i]} g^r$, $S_2 = S'_2 g^a$ and $S_1 = (u'M')^a S'_1/(S'_2)^r$ (this only holds when S'_1, S'_2 are correctly computed by the signer for M' yet if the signer does not, the user can detect this misbehavior and abort).

At the same time, there exists another set of randomness $(\bar{r}, \bar{a})$ such that $M' = \prod_{i=1}^{n} u_i^{m[i]} g^{\bar{r}}$, $S_2 = \bar{S}'_2 g^{\bar{a}}$ and $S_1 = (u'M')^{\bar{a}} \bar{S}'_1/(\bar{S}'_2)^{\bar{r}}$ (again, assume S'_1 and S'_2 are correctly computed).

This means that a message-signature pair can be the result of interaction M', S'_1, S'_2 or $\bar{M}', \bar{S}'_1, \bar{S}'_2$. Finally, due to the simulatablility of $\mathfrak{P}_{\mathsf{WH}}$, $\pi_{M'}$ nor $\pi_{\bar{M}'}$ leaks no information about m and r under the DLIN and CDH assumption. Thus, our construction possesses blindness under the DLIN and CDH assumption.